We are running a program, and when we look at its entry in **/proc**, we see:

```
00010000-00011000 r-xp 00000000 00:22 126617690  /home/plank/a.out
00020000-00021000 rw-p 00000000 00:22 126617690  /home/plank/a.out
012cb000-012ec000 rw-p 00000000 00:00 0          [heap]
76f3a000-76f3b000 r--p 0001f000 b3:07 524429     /lib/arm-linux-gnueabihf/ld-2.19.so
76f3b000-76f3c000 rw-p 00020000 b3:07 524429     /lib/arm-linux-gnueabihf/ld-2.19.so
7ebc3000-7ebe4000 rwxp 00000000 00:00 0          [stack]
```

The program prints the return value of **getpagesize()**, and it is 4096 (0x1000).

Question 1: Your program has a procedure called **proc()**. What is the smallest address greater than or equal to **proc** that will segfault when you try to read it? You can give an absolute address or a value relative to **proc**.

Question 2: What is the smallest address greater than or equal to **proc** that will segfault when you try to write it? You can give an absolute address or a value relative to **proc**.

Question 3: Can **&edata** be equal to 0x10f00? (Answer Yes or No).

Question 4: Can **&etext** be equal to 0x10f00? (Answer Yes or No).

Question 5: How many pages are in the heap (you can answer in hex or decimal)?

Question 6: If I make enough procedure calls, the beginning of the stack, as reported by the entry in **/proc**, will be changed to some number smaller than 0x7ebc3000. (Answer T or F).

Answers to Clicker Questions

- Question 1: 0x11000 -- this is the first address on the next page.
- Question 2: proc. Since the code (text) segment disallows writes, trying to write any address in the code segment will cause a segmentation violation.
- Question 3: No -- **&edata** is in the globals (data) segment, which is between 0x20000 and 0x21000. Note it is protected as read/write.
- Question 4: Yes -- **&etext** is in the code (text) segment.
- Question 5: The addresses of the heap are 0x12cb000-0x12ec000. That means 0x12ec-0x12cb = 0x21 pages. In decimal, that's 33.
- Question 6: False. The stack doesn't change in size. Instead, you will eventually segfault when you set the stack pointer to an address that is too low.